Universidade de Caxias do Sul

Área do Conhecimento de Ciências Exatas e Engenharias

Disciplina de Fundamentos de Sistemas Operacionais

Professor Daniel Luis Notari

Avaliação I – 22/05/2022

Aluno: Eduardo Eberhardt Pereira

1,5

Questão 1. (1,5) Diferentes algoritmos de escalonamento de CPU possuem diferentes propriedades e a escolha de um determinado algoritmo pode favorecer uma classe dos processos em detrimento de outra. Explique como funciona o algoritmo de Prioridades com Múltiplas filas com realimentação. Desenhe um exemplo.

1,0

Questão 2. (1,5) Um processo passa por uma série de estados de processo distinto. Em relação aos estados de transição de processos, explique o funcionamento das transições de estado:

a) Quando um processo é despachado, ele transita de "pronto" para "em execução".

b) Quando um processo é bloqueado, ele transita de "em execução" para "bloqueado".

c) Quando um processo é considerado "zumbi" no Linux

1,5

Questão 3. (1,5) Os Sistemas Operacionais líderes de mercados são Windows e Linux. Cite e explique os tipos de estrutura de sistemas operacionais que eles usam.

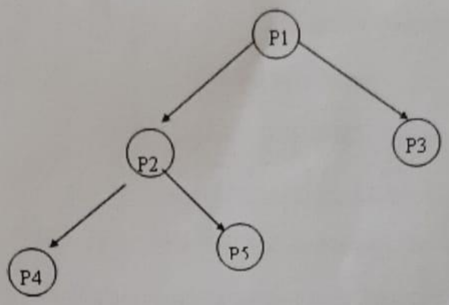
1,2

Questão 4. (1,5) Considere o conjunto de 4 processos abaixo, onde a chegada corresponde ao momento em que o processo foi posto a primeira vez no estado de apto e, o tempo de CPU é a quantidade de processamento necessário. Assumindo que a execução inicia no tempo 0 e, que a troca de contexto seja feita instantaneamente, ou seja, com um custo zero, determine o tempo médio de espera para uma política FCFS e Round Robin. OBS: considerar um quantum de 5 unidades.

Processo	Tempo de CPU
P0	15
P1	25
P2	20
P3	15

2,0

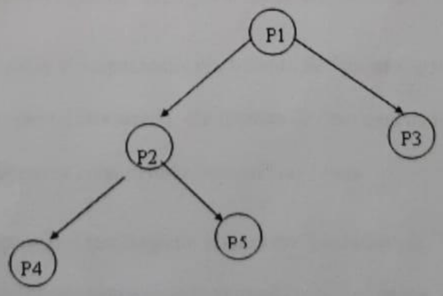
Questão 5. (2,0) Escreva um código correspondente ao grafo de hierarquia de processos abaixo utilizando a chamada de sistemas *fork*.



1,5

Questão 6. (2,0) Escreva um código correspondente ao grafo de hierarquia de processos abaixo utilizando a chamada de sistemas **de pthreads com as primitivas de exclusão mútua.**

2,0



Eduardo Eberhardt Pereira

1) O algoritmo de prioridades com múltiplas filas com realimentação utiliza de um ordenamento baseado em diferentes níveis de prioridade, permitindo que o execute primeiro os processos mais importantes. Essa variação utiliza de múltiplas filas distintas para evitar starvation, que é quando um processo de baixa prioridade não consegue ser executado por sempre existirem processos de maior prioridade que ele. Com múltiplas filas, até os processos com menor prioridade recebem tempo de CPU. A realimentação faz com que as filas sejam reorganizadas baseado nos processos que as compõem, processando a entrada de novos processos e na dinamismo das prioridades.

EX.

entrada

FILA 1: → [P0 prioridade=5] [P4 prioridade=7] [P3 prioridade=7] saída →

FILA 2: → [P1 prioridade=0] [P5 prioridade=2] [P2 prioridade=2] saída →

2) a) Quando o escalonador define que um processo na fila de prontos deve ser executado, ele realiza a troca de processos, salvando o estado do atual e recuperando o estado do selecionado. Assim, o processo que estava pronto agora está executando.

b) Quando um processo está esperando o término de outro, esperando uma entrada de algum dispositivo de I/O (input/output) ou esperando algum outro dado, sua execução é interrompida e o processo é considerado bloqueado até receber o dado que espera.

c) Um processo é considerado zumbi quando ele provoca deadlock no escalonador. Isto é, o escalonador não consegue decidir como proceder pois dois processos estão, em geral, esperando um ao outro. Como ambos esperam um ao outro, a espera nunca termina e os processos ficam como zumbis, esperando para sempre.

3) As estruturas e arquiteturas dos sistemas operacionais Windows e Linux são as mesmas, e se baseiam em:

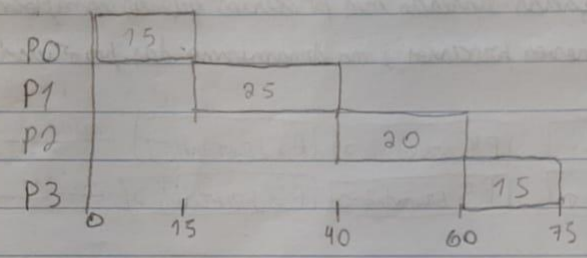
- monolítica: sem estrutura ou módulos, qualquer rotina do SO pode chamar outra (embora o usuário tenha restrições)

continua →

35
-30
30
-05
50

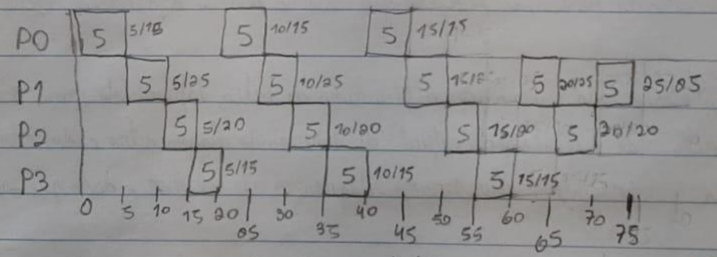
- em camadas: possui diferentes camadas com diferentes níveis de acesso, e faz uso de requisições (traps) para acessar camadas mais internas.
- máquinas virtuais: um sistema operacional é um processo em outro, permitindo retrocompatibilidade.

4) FCFS:



Tempo de Espera: (0+15+40+60)/4 = 28,75 u.t. C

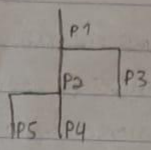
ROUND ROBIN: (quantum = 5)



Tempo de Espera: (0+5+10+15)/4 = 6 u.t. X

7,5 C

```
void main() {
5) pid_t pid_1, pid_2;
   P1();
   pid_1 = fork();
   if (pid_1 > 0) {
      P2();
      pid_2 = fork();
      if (pid_2 > 0) {
         P4();
      } else {
         P5();
      }
   } else {
      P3();
   }
}
```




```
int vetor[5]; int P1=0, P2=1, P3=2, P4=3, P5=4;
6) pthread_mutex_t mutex;
   void main() {
      pthread_t tid_1, tid_2, tid_3, tid_4, tid_5;
      pthread_mutex_init(&mutex, NULL);
      for(int i=0; i<5; i++) vetor[i] = i+1;
      pthread_create(&tid_1, NULL, imprime_nodo, (void*)P1);
      pthread_join(tid_1, NULL);
      pthread_create(&tid_2, NULL, imprime_nodo, (void*)P2);
      pthread_create(&tid_3, NULL, imprime_nodo, (void*)P3);
      pthread_join(tid_2, NULL);
      pthread_create(&tid_4, NULL, imprime_nodo, (void*)P4);
      pthread_create(&tid_5, NULL, imprime_nodo, (void*)P5);
      pthread_join(tid_3, NULL);
      pthread_join(tid_4, NULL);
      pthread_join(tid_5, NULL); }
```

imprime_nodo na prox folha →